1. 開發環境:Windows11 anaconda python 3.8(需安裝程式裡面含有的package才能正常的執行)
2. 實作方法和流程、包含3.特殊機制考量與設計:



有設計一個讀檔和寫檔的class

在任務一和二有設計一個進度條，因為任務一的方式真的太慢了，需要知道還有多久，還有必須等待進度條跑完再去輸入下一筆input輸入檔名資料，不然會導致進度條出錯。

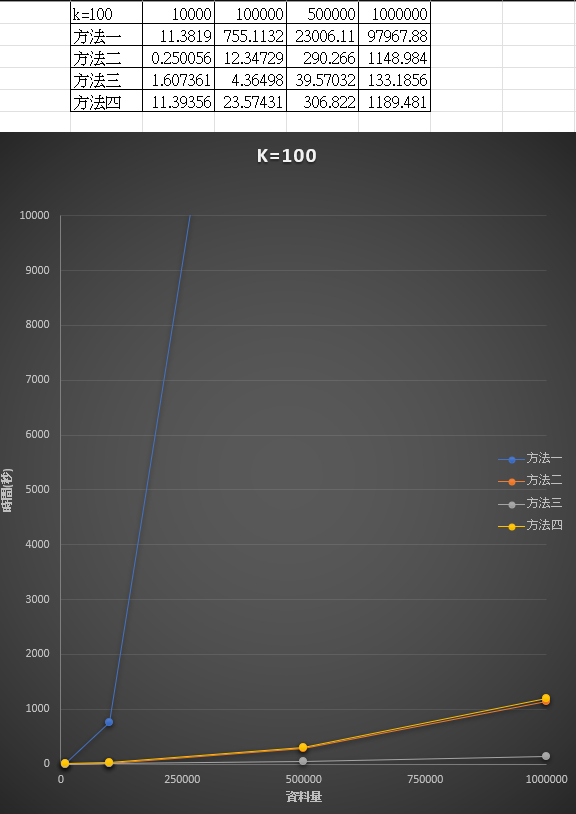
程式正在執行請耐心等候，尤其是任務一、二、四

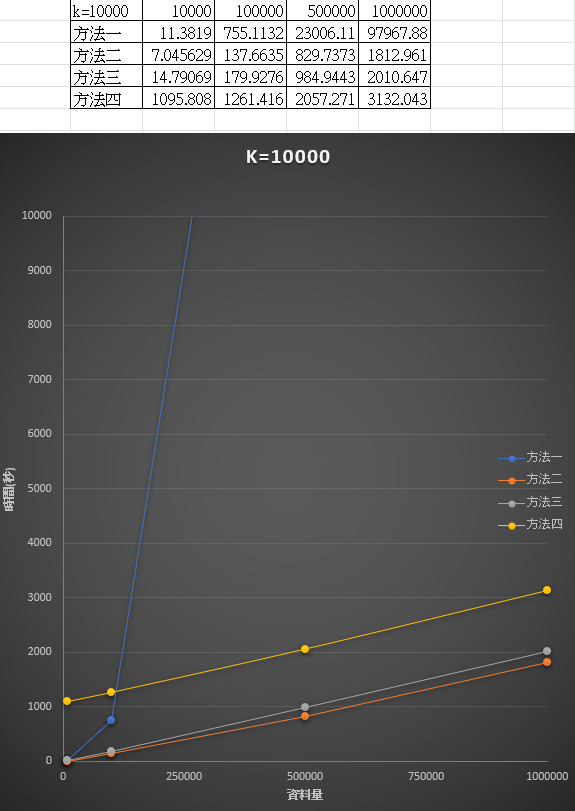
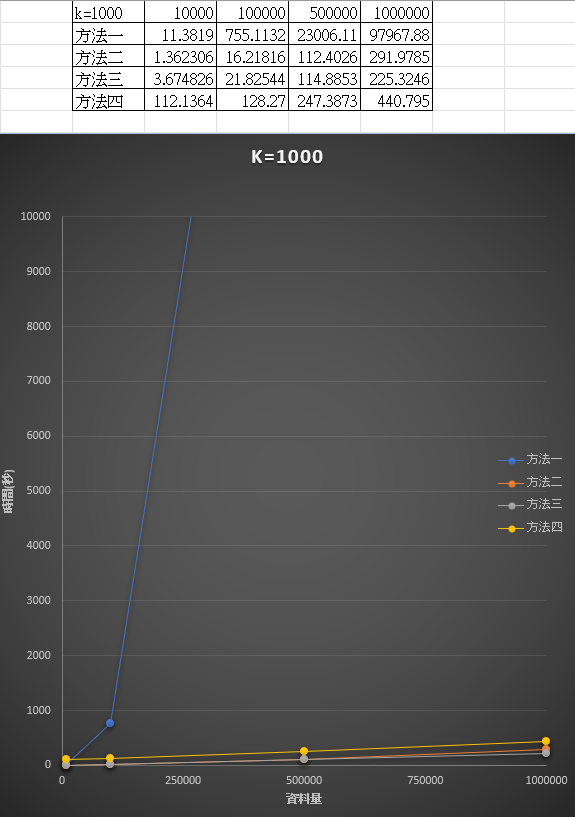
1. 第一題對於讀檔好的資料(List)，就直接使用兩層while迴圈進行bubble sort
2. 第二題先把讀檔好的資料(List)，利用在一個process內對於切好的k份資料，利用迭代的方式做完k 次bubble sort之後，再做k-1次的merge
3. 第三題先把讀檔好的資料(List)，我是利用multiprocessing package 裡面pool的功能，用非同步的方式對於切好k分的資料進行bubble sort之後，能很好的運用電腦CPU的效能，假如k值大於或等於CPU的核心數量，bubble sort的速度會很快，而merge是用starmap的funciton，可能沒辦法好好利用CPU多核心的效能，因為在這邊我的寫法是兩個sort好的資料，merge到一個buffer裡面，每個sort好的資料都必須等前面的merge結束才能繼續merge，而且merge對於資料量大的時候和k值大的時候速度相對會慢許多，不過也是4種方法裡在k值為100最快的
4. 第四題先把讀檔好的資料(List)，我自己引用了threading的package寫一個Return value of Thread的物件Return value of Thread的物件取得thread裡面執行完return的資料，一開始是先建立k個Return value of Thread的物件，裡面就包含了thread物件，好處是可以交由這k個thread物件進行bubble sort，再交由這k個thread物件進行merge sort，這個方法在k值很大的時候有不錯的效果，能產生很多thread幫忙bubble sort，也因我的寫法bubble sort的交於k個thread執行沒有資料共用的問題，但merge卻需要等前面的thread用完才行，所以相對於multi thread，我的寫法沒有能好好改善利用多的thread去幫忙merge，因為需要上一個thread merge完的資料，為了thread之間的互斥性，所以我的方法是讓後面的thread等前面的thread。
5. 分析結果和原因:

[不同 N 值 vs. 執行時間]比較四種方法在資料筆數 1 萬、10 萬、50 萬、100 萬所耗費的時間，(共 16 次實驗，如下表格)，執行速度的 差別，使用圖表分析。

1. 任務二三四k值分析

K的大小必須要找到一個平衡點，不能大也不能小，k值太大在我的方法裡會使merge的時間拉長，但bubble sort的時間變短，太大或太小會有一好沒兩好的情況，在二三四的任務中都有一個很特別的問題，就是k太大的時候，速度會變得很慢很慢，原因是k越大，merge的次數變多，k值為100的時候，速度卡在bubble sort上面，k值為10000的時候，速度卡在merge上面，在k值為1000的時候可以看到bubble sort 和 merge取得良好的平衡。





|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |